# Advanced Computer Systems Project 1 Report

Ivan Cheng (ECSE)

Rensselaer Polytechnic Institute

ECSE 4320 Advanced Computer Systems

Dr. Tong. Zhang

October 1, 2025

## Setup/methodology

## Hardware:

The experiments were performed on an Intel Core i7-11850H laptop. The laptop runs on Tiger Lake-H, 8 cores/16 threads, 2.5 Ghz base. Cache hierarchy includes 32 KB L1 data cache per core, 1.25 MB L2 per core, and a shared 24 MB L3 cache.

According to lscpu the processor supports SIMD extensions up to AVX-512. Compiler vectorization reports confirmed that the auto-vectorized builds use 32-byte (AVX2) vector instructions which is equivalent to 8 float32 operations per vector. AVX2 was the ISA utilized in this project for stability.

## Software:

**Operating System:** Ubuntu 20.04 running under Windows.
**Compiler:** GNU g++ version 9.3.0.
**Compiler Flags:**
**Scalar Baseline Build:** g++ -O3 -fno-tree-vectorize project1.cpp -o project1_scalar
**SIMD build:**
g++ -O3 -march=icelake-client -fopt-info-vec-optimized project1.cpp -o project1_simd
**Frequency Policy:**
Experiments were run with the default Linux CPU governor under WSL2, which adjusts frequency For reporting, cycles per element (CPE) were estimated using a fixed nominal frequency of 3.0 GHz to normalize across runs. Due to base frequency of the i7-11850H being 2.5GHz and turbo boosting up to 4.8GHz depending on power/thermal conditions. This avoids skew from transient turbo boosts or thermal throttling.
**SMT State:**
All experiments were executed in a single thread to isolate SIMD effects. SMT was enabled on the CPU but only one thread was used in each run.
**Measurement Method:**
**Scalar build: g++ -O3 -fno-tree-vectorize project1.cpp -o project1_scalar**
**SIMD build: g++ -O3 -std=c++17 -march=core-avx2 -fopt-info-vec-optimized project1.cpp -o project1_simd**
Execution time was measured with std::chrono::high_resolution_clock. Each kernel was executed for 5 repetitions per problem size, and the mean runtime was reported.

- GFLOP/s = FLOPs per kernel ÷ runtime.
- Cycles per element (CPE) = (runtime × clock frequency) ÷ problem size.
- Speedup = scalar runtime ÷ SIMD runtime.

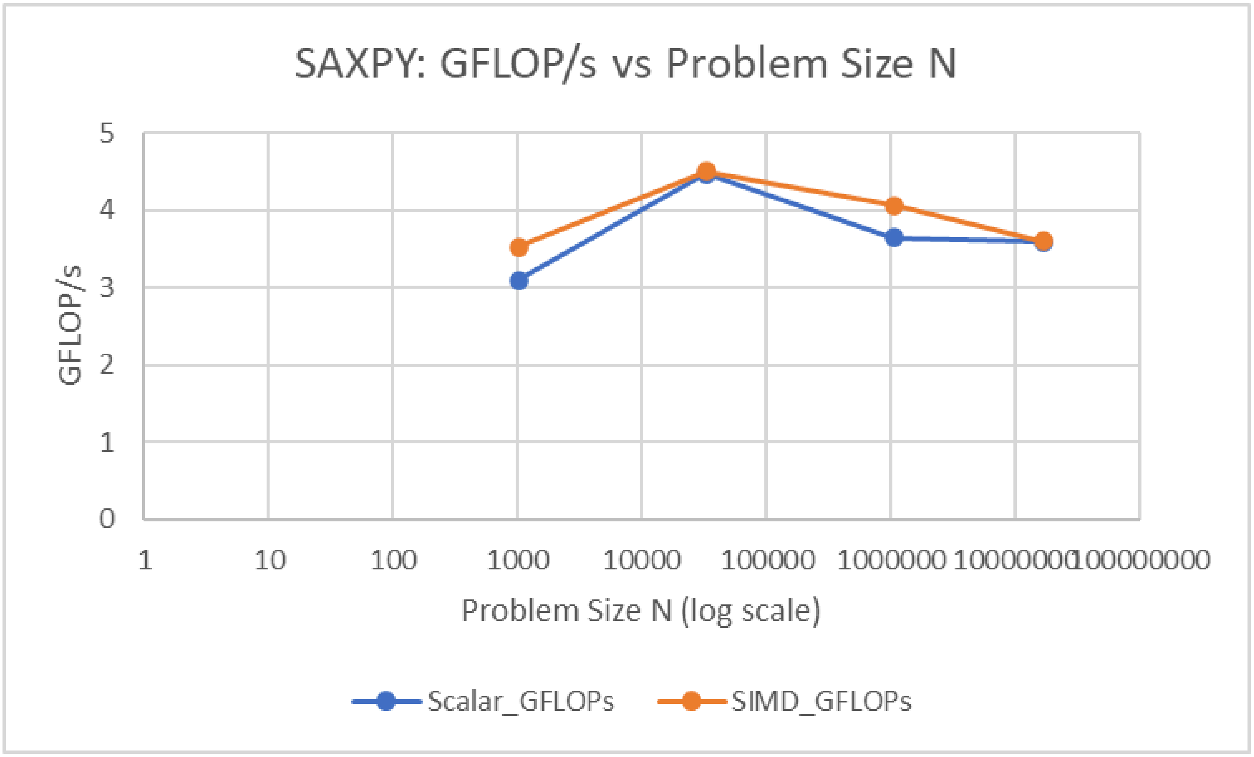
**Data Initialization:**
Vectors were initialized with pseudorandom floats in the range [0, 1] using std::mt19937. Correctness was validated by comparing outputs with a tolerance of 1e-6.

## Plots/Tables

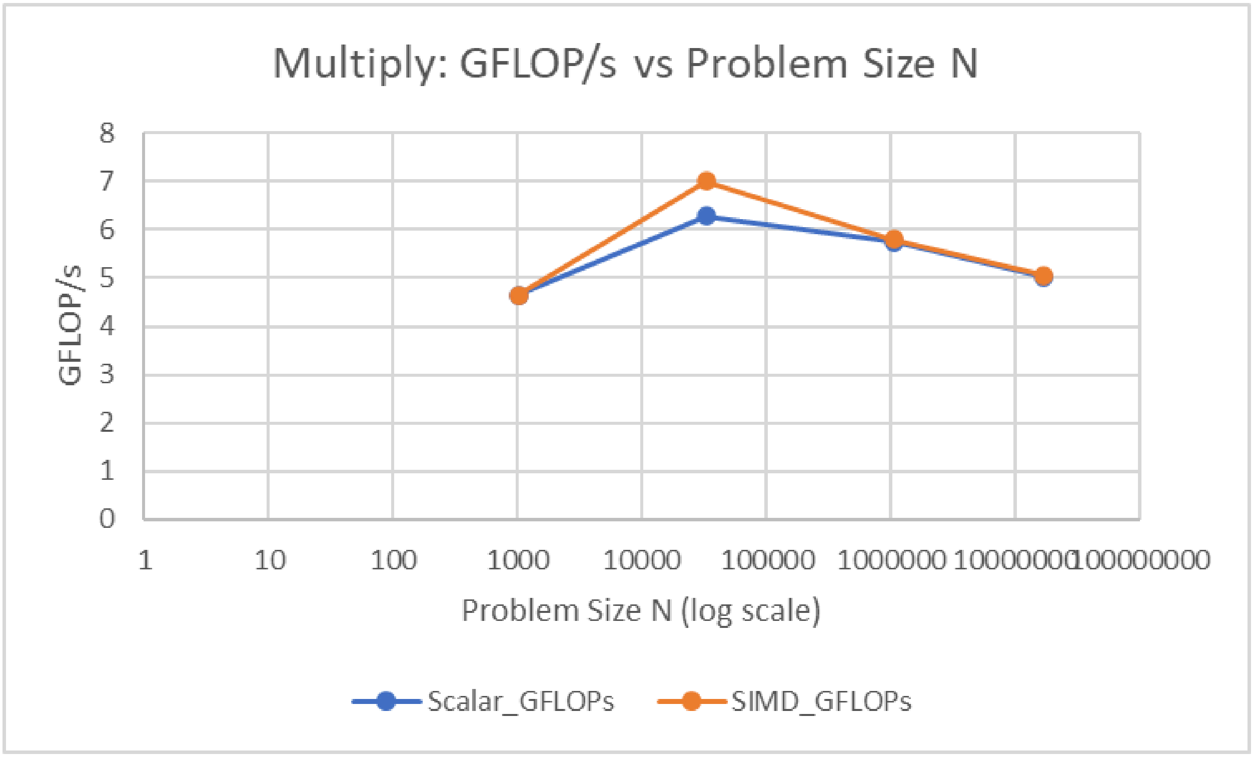
| N | Kernel | Scalar_GFLOPs | SIMD_GFLOPs | Speedup | Scalar_CPE | SIMD_CPE |
|---|---|---|---|---|---|---|
| 1024 | SAXPY | 3.10303 | 3.53103 | 1.13793 | 1.93359 | 1.69922 |
| 1024 | Dot | 1.21905 | 1.21905 | 1 | 4.92187 | 4.92187 |
| 1024 | Multiply | 4.65455 | 4.65455 | 1 | 0.644531 | 0.644531 |
| 32768 | SAXPY | 4.4765 | 4.5011 | 1.00549 | 1.34033 | 1.33301 |
| 32768 | Dot | 1.24593 | 1.24593 | 1 | 4.81567 | 4.81567 |
| 32768 | Multiply | 6.27739 | 7.00171 | 1.11538 | 0.477905 | 0.428467 |
| 1048576 | SAXPY | 3.64671 | 4.06488 | 1.11467 | 1.64532 | 1.47606 |
| 1048576 | Dot | 1.20631 | 1.2027 | 0.997006 | 4.97383 | 4.98877 |
| 1048576 | Multiply | 5.75508 | 5.7958 | 1.00707 | 0.521278 | 0.517616 |
| 16777216 | SAXPY | 3.59475 | 3.60226 | 1.00209 | 1.6691 | 1.66562 |
| 16777216 | Dot | 1.18279 | 1.19437 | 1.00979 | 5.07277 | 5.02359 |
| 16777216 | Multiply | 5.01723 | 5.05344 | 1.00722 | 0.597939 | 0.593655 |

## GFLOP/s vs N

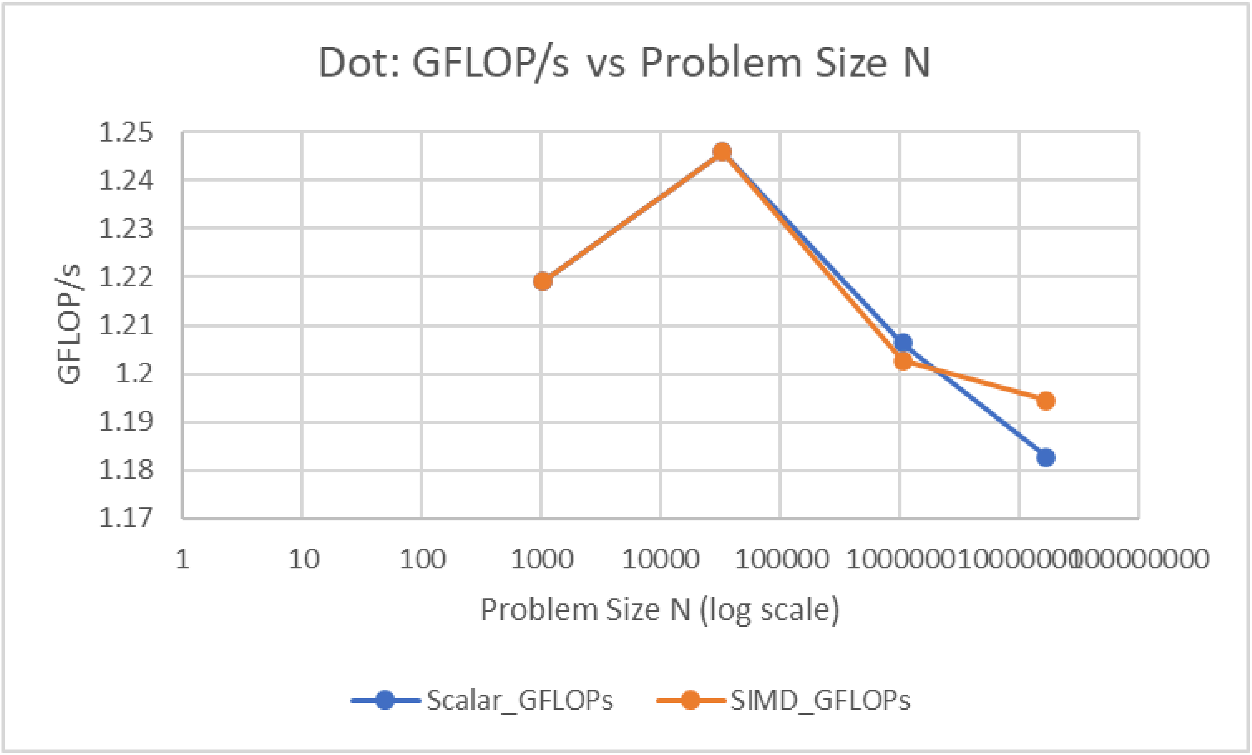
SAXPY: GFLOP/s vs Problem Size N



Multiply: GFLOP/s vs Problem Size N



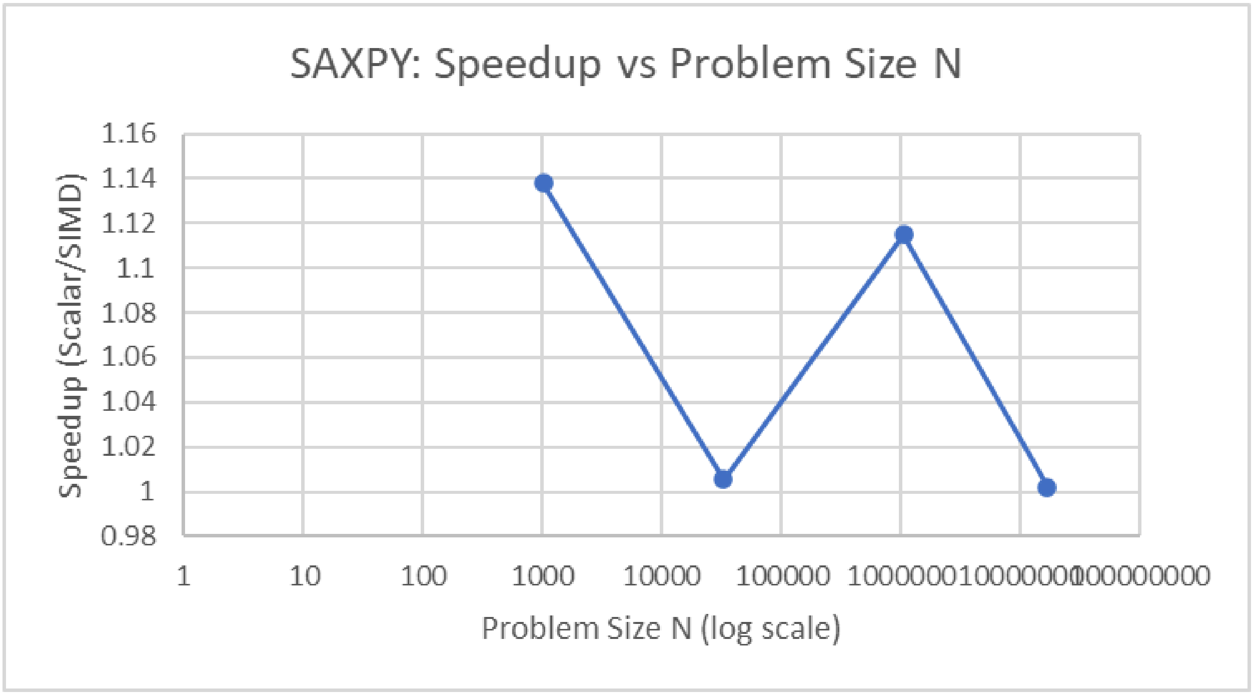
Dot: GFLOP/s vs Problem Size N



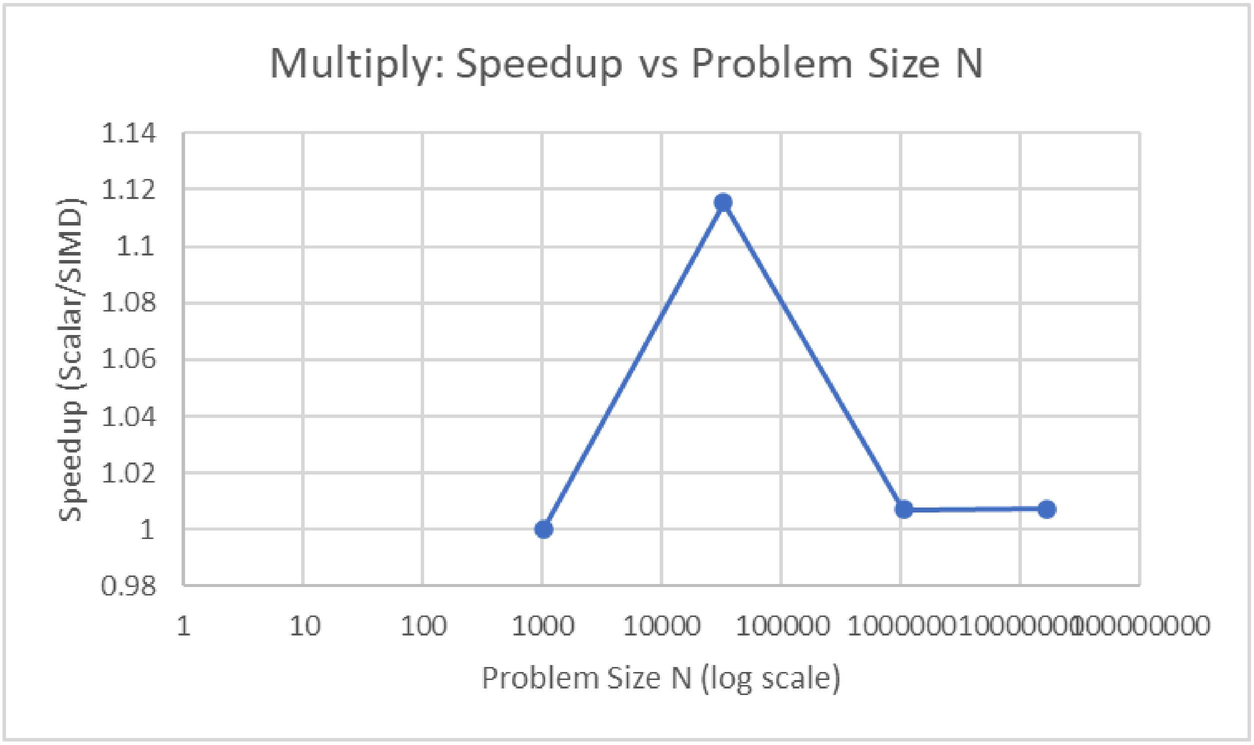
For float kernels, the cache cutoffs occur at N≈4K (L1, 32 KB), N≈32K (L2, 256 KB), and N≈1.5M (L3, 12 MB). Performance drops in the GFLOPs vs N plots at the cache cutoffs, with further degradation beyond L3 when data no longer fits in cache and accesses spill to DRAM.

## Speedup vs N

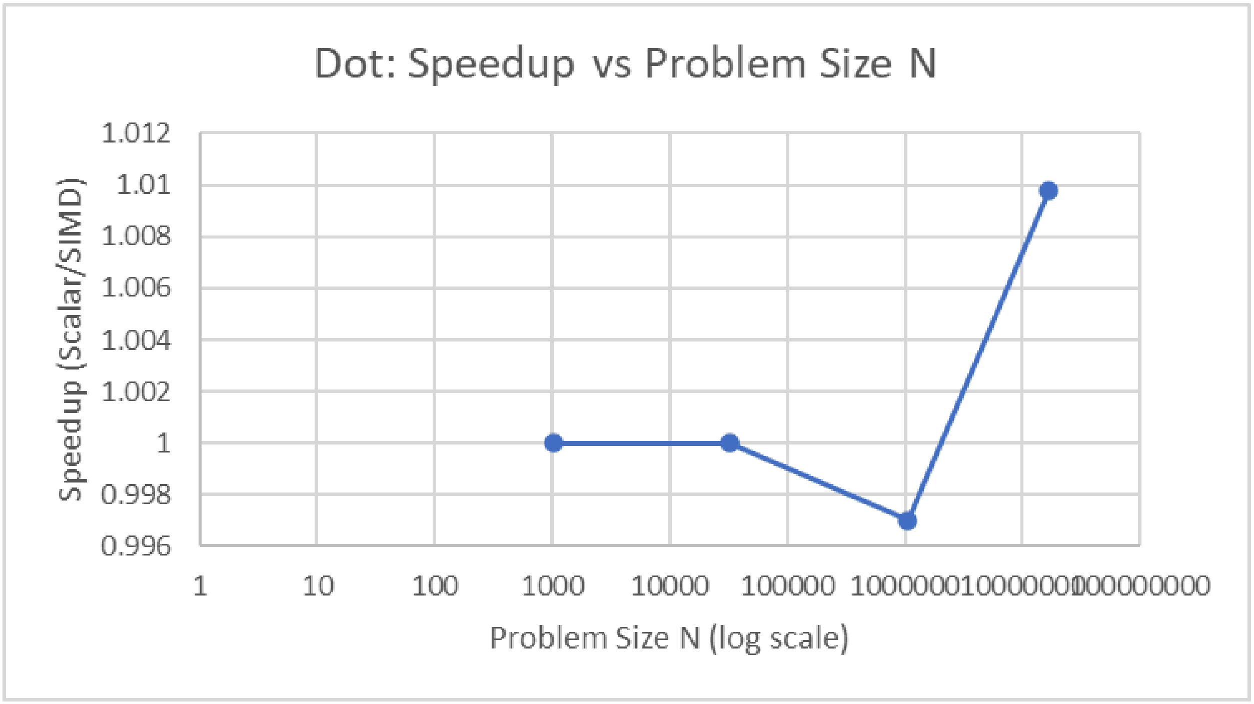
SAXPY: Speedup vs Problem Size N



Multiply: Speedup vs Problem Size N

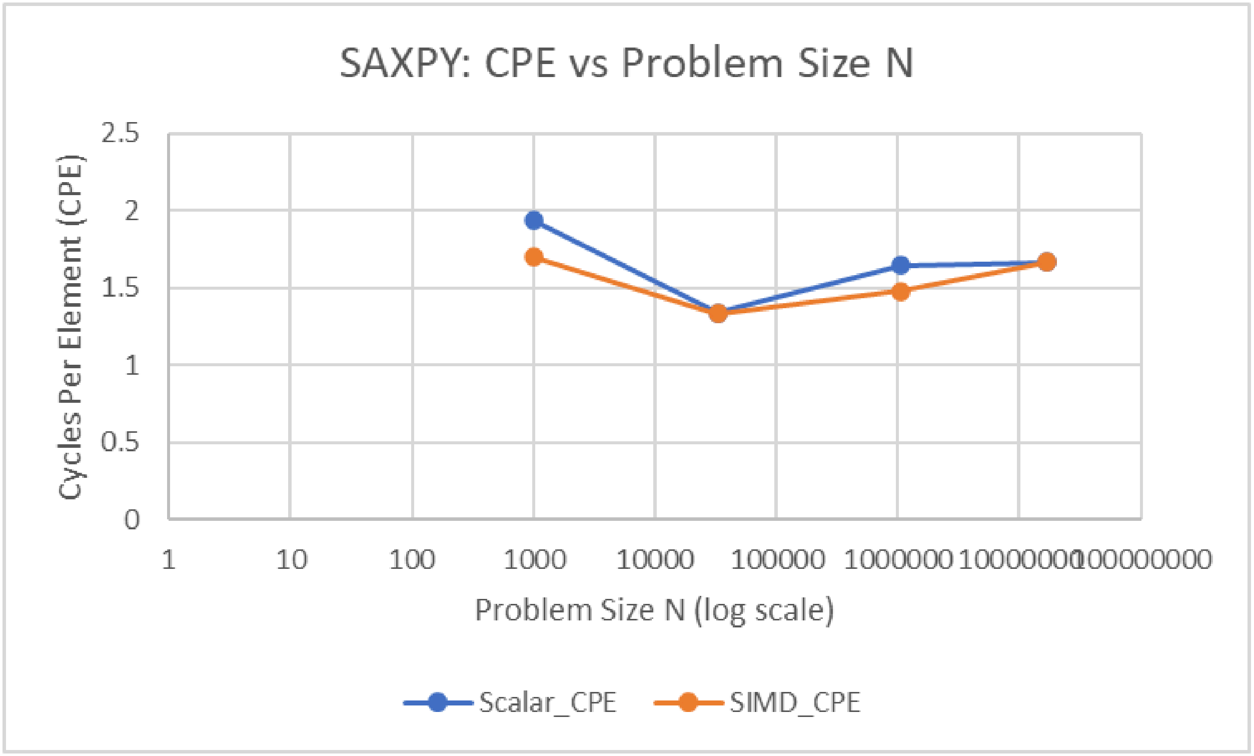


Dot: Speedup vs Problem Size N

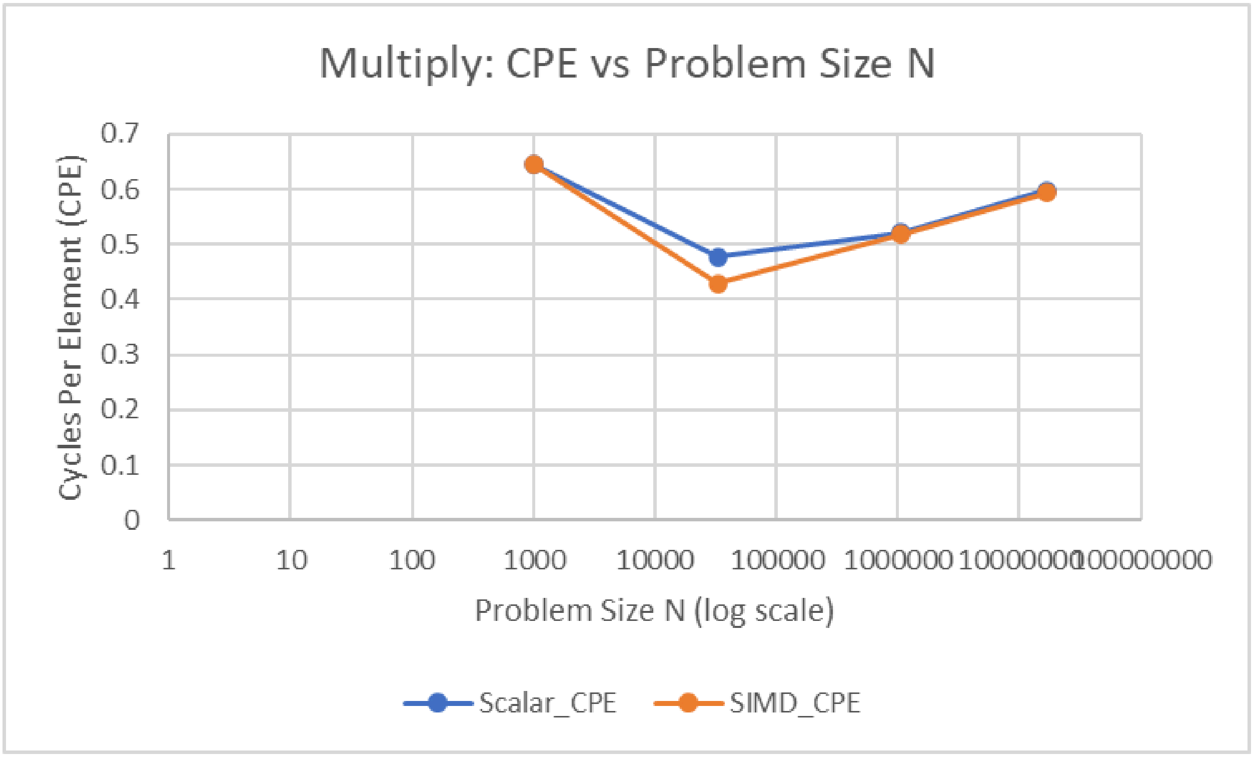


## Cycles per element vs N

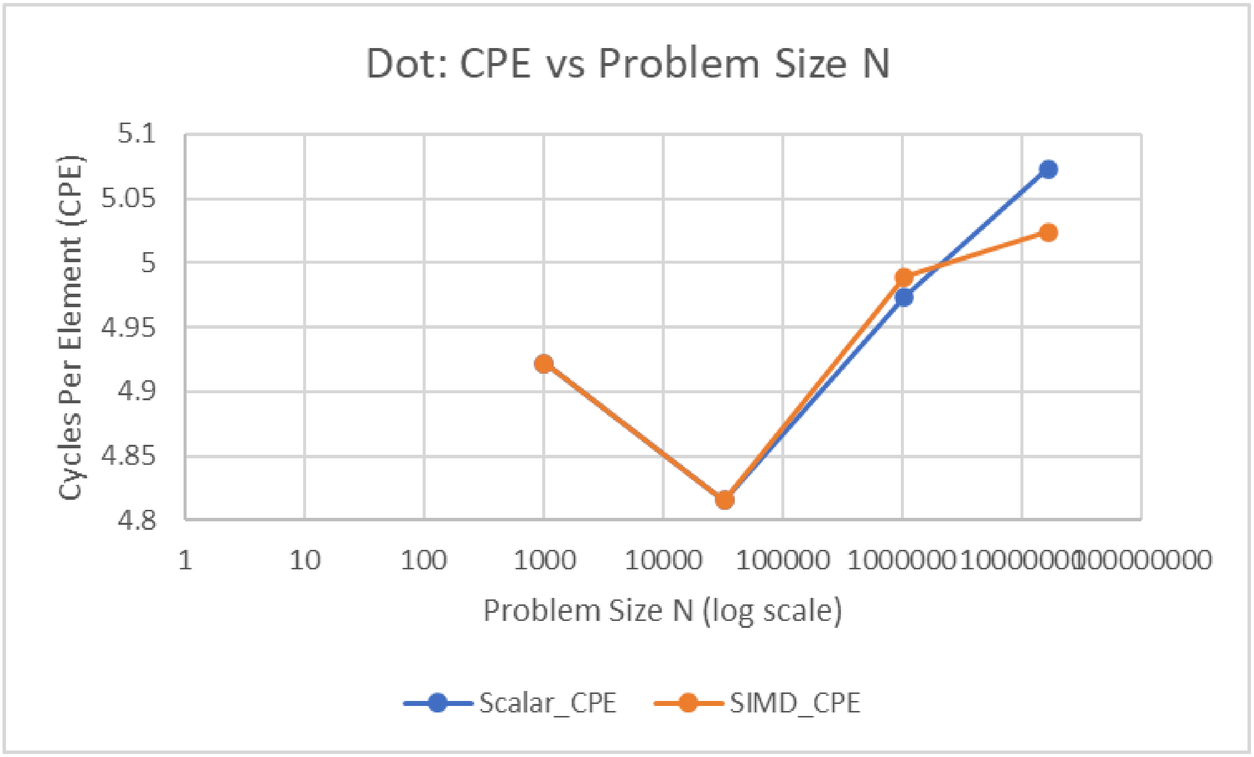
SAXPY: CPE vs Problem Size N



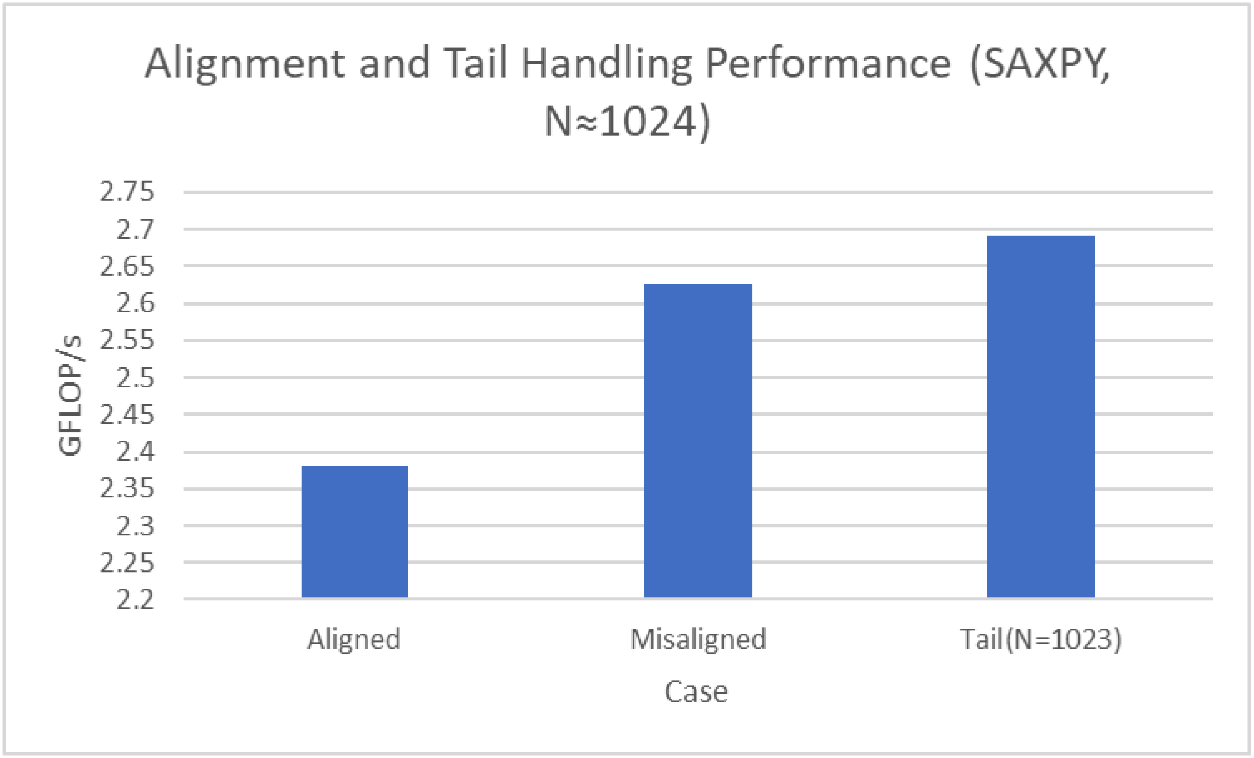
Multiply:CPE vs Problem Size N



Dot: CPE vs Problem Size N

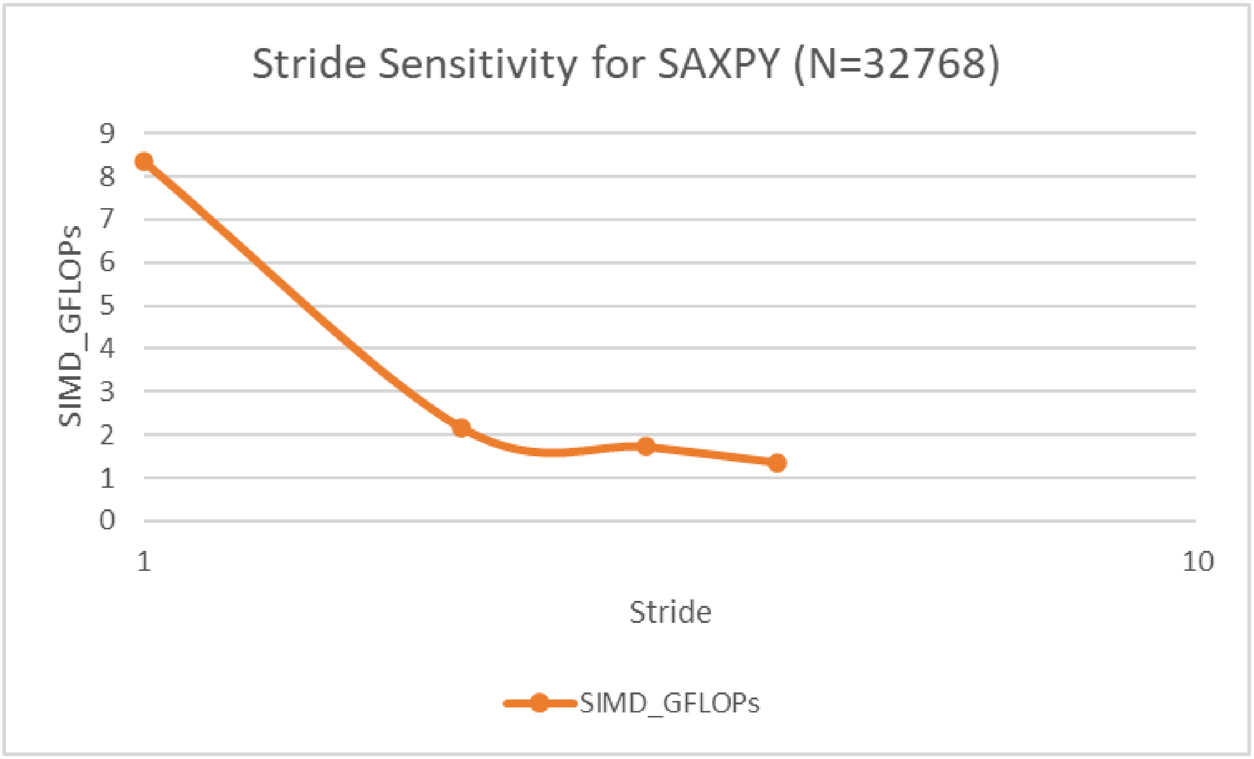


## Alignment & Tail-handling



The chart above compares SAXPY performance across three cases: aligned arrays, misaligned, and a non-divisible size. Aligned accesses achieve the highest performance, while misaligned reduces efficiency. The tail case also lowers efficiency, as the compiler must generate scalar cleanup code for the elements not divisible by the SIMD width.

## Stride Effects



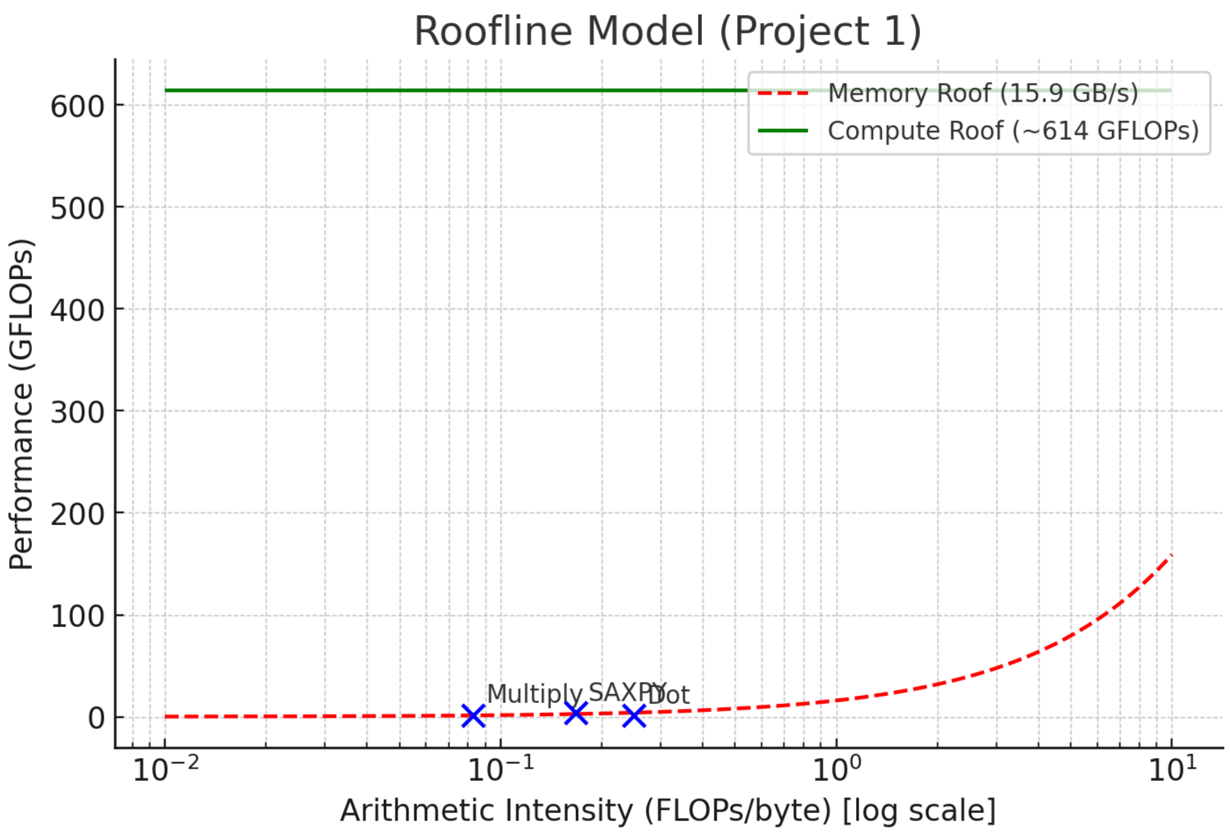
The chart above shows the stride sensitivity for SAXPY (N=32768). Performance drops sharply as stride increases due to reduced cache line utilization. With stride=1, each cache line is fully used, while stride=8 wastes most of the fetched data, resulting in memory-bandwidth inefficiency.

## Data Type Comparison

| Kernel | N | Precision | SIMD GFLOP/s | Scalar GFLOP/s | Speedup |
|---|---|---|---|---|---|
| Multiply | 1,048,576 | Float32 | 2.2093 | 1.8650 | 1.18x |
| Multiply | 1,048,576 | Float64 | 0.7524 | 0.7687 | 0.98x |

When comparing Float32 and Float64 SIMD achieved a 1.18x speedup over scalar in float32 whereas in float64 there was essentially no speedup. The reason for this may be due to the experiment being run in AVX2 where each vector register holds 8 floats compared to 4 doubles which halves the potential throughput which explains the lower performance.

## Roofline Interpretation



In the chart above we used the SAXPY kernel experiment for our plot. From our SIMD results (stride = 1, N= 16M) Saxpy achieved 3.61 GFLOP/s. The Arithmetic Intensity measured was 0.167 FLOPs/byte. Roofline bounds were measured from project 2 to be 15.9 GB/s. The maximum attainable performance is 2.65 GFLOP/s. The Compute roof for my CPU Intel i7-11850H is 614 GFLOP/s. Our achieved performance was a bit higher than predicted bandwidth but way below the compute roof of the CPU. SAXPY lies on the sloped memory roof which shows its memory bound.

## Analysis

The SAXPY kernel demonstrated modest SIMD speedup at cache-resident sizes, with performance 4-4.5 GFLOP/s. SIMD achieved around ~1.00-1.14x speedup when vectors fit in L1/L2 or LLC. However, once the working set exceeded cache and became DRAM-resident, the scalar and SIMD performance tended to become similar. CPE became near 1.5-1.7 CPE. This matches what is expected due to SAXPY's low arithmetic intensity which makes it strongly memory bound with limited benefits from SIMD outside of the cache.

The multiply kernel reached a higher peak throughput of around 7 GFLOP/s in cache-resident sizes, where SIMD achieved its best gains. The maximum SIMD speedup was ~1.00-1.12x, in the L1/L2 and LLC regions. Once the problem size exceeded cache and into DRAM, the performance of scalar and SIMD tended to become similar. CPE flattened around ~0.5-0.6 CPE. These are slightly expected results as multiply kernel has a slightly higher arithmetic intensity compared to SAXPY. In theory the speedup should be greater however they are similar and this may be due to my compiler implementation or measurement limits.

The dot product kernel had the lowest peak, ~1.1-1.2 GFLOP/s, with SIMD showing almost no advantage. The maximum speedup was ~1.00x in cache-resident ranges, where scalar and SIMD had similar performance. At large N, speedup is shown to rise to maximum ~1.01x likely from the compiler vectorizing multiples. This is somewhat expected because the kernel remains largely memory and dependency bound with performance between scalar and SIMD being similar. However, in theory it is slightly different as the maximum achievable speedup should be 1x which is very slightly off from the 1.01x result I achieved.

All kernels (SAXPY, Multiply, Dot) produced identical outputs between scalar and SIMD versions within a tolerance of 1e-6, confirming numerical correctness

## Limitations/Anomalies

Several minor anomalies were observed across experiments. At small problem sizes, results showed variability due to timer resolution, loop setups, and cache warm-ups. Mid range sizes, SIMD performance being slightly lower than scalar. At large problem sizes scalar and SIMD performance generally were similar as expected since memory bandwidth became the limiting factor. In addition, external factors such as OS background activity and WSL activity may have influenced measurements.

## Reference

[1] "Intel® CoreTM i7-11850H Processor (24M Cache, up to 4.80 GHz)," Intel, 2025. https://www.intel.com/content/www/us/en/products/sku/213799/intel-core-i711850h-processor-24m-cache-up-to-4-80-ghz/specifications.html?wapkw=i7-11850H
[2] OpenAI, "ChatGPT," ChatGPT, 2025. https://chatgpt.com/